MLP neural network based gas classification system on Zynq SoC

MLP Neural Network based Gas Classification System on Zynq SoC

Xiaojun Zhai, Amine Ait-Si-Ali, *Student Member, IEEE*, Abbes Amira, *Senior Member, IEEE* and Faycal Bensaali, *Senior Member, IEEE*

***Abstract* — Systems based on Wireless Gas Sensor Networks (WGSN) offer a powerful tool to observe and analyse data in complex environments over long monitoring periods. Since the reliability of sensors is very important in those systems, gas classification is a critical process within the gas safety precautions. A gas classification system has to react fast in order to take essential actions in case of fault detection. This paper proposes a low latency real-time gas classification service system, which uses a Multi-Layer Perceptron (MLP) Artificial Neural Network (ANN) to detect and classify the gas sensor data. An accurate MLP is developed to work with the data set obtained from an array of tin oxide (SnO2) gas sensor, based on convex Micro hotplates (MHP). The overall system acquires the gas sensor data through RFID, and processes the sensor data with the proposed MLP classifier implemented on a System on Chip (SoC) platform from Xilinx. Hardware implementation of the classifier is optimized to achieve very low latency for real-time application. The proposed architecture has been implemented on a ZYNQ SoC using fixed-point format and achieved results have shown that an accuracy of 97.4% has been obtained.**

***Index Terms*— Artificial Neural Network; Gas identification; FPGA; System on Chip (SoC); ZYNQ.**

I. INTRODUCTION

The oil and gas industry is one of the most dominant industries for the application of Wireless Sensor Technology [1]. For gas application, Wireless Gas Sensor Networks (WGSN) systems are used to observe and analyse sensed gas data in complex environments over long periods. As the accuracy of data and reliability of sensors are very important in those systems, a gas classification system has to react fast in order to take essential actions in case of fault detection. In order to have a system that reacts fast to the changes each processing element within the system has to work in a low latency manner [2]. Generally classifiers within the gas sensor array system are the most computationally intensive processing components. Classifiers, such as neural networks, use the data set acquired from the gas sensor array system and process them to detect and classify gases and their properties within the gas chamber.

Feed-forward artificial neural networks (ANN) are commonly used as classifiers for pattern classification approaches [3], which also include multi-layer perceptron (MLP). In general, software implementation of the MLP neural networks are used during the algorithm development phase, where parallel and low latency approach is not needed. However in real-world applications high speed processing and low latencies are needed in order to execute the ANN within the real-time constraints.

The ubiquitous nature and miniaturization of sensor devices have revolutionized surveillance and monitoring systems. Now a single sensor node can be equipped with multiple sensors to collect data from different modularities. Since sensor nodes are becoming increasingly accurate they are employed to monitor subtle changes to the environment. In addition, machine learning techniques can be used to identify behavioural changes by analysing the data collected from sensor nodes, and generate an alarm signal that indicates abnormal behaviour.

In this paper, a low-latency, real-time gas classification system is proposed. The service system uses a MLP ANN to detect and classify the gas sensor data. An accurate MLP is developed to work with the data set obtained from an array of tin-oxide (SnO2) gas sensor [2], based on convex Micro hotplates (MHP). The proposed system acquires the gas sensor data through RFID, and processes the sensor data with a novel low latency classifier within a heterogeneous ZYNQ System-on-Chip (SoC) platform from Xilinx. ARM processor within the ZYNQ SoC acts as a host-processing platform that handles the data acquisition and transmission as well as the data distribution to and from the classifier. The interaction between the Processing System (PS) and the programmable logic (PL) is done with the use of the Direct Memory Access (DMA) accesses within the ZYNQ platform. The use of DMA engine provides the classifier the required bandwidth to be able to give the throughput required for the overall system. Finally hardware implementation of the MLP classifier is optimized to have very low latency and response time, in order to process

This paper was submitted for the review on 26th September, 2016. This research work was supported by National Priorities Research Program (NPRP) grant No. 5-080-2-028 from the Qatar National Research Fund (a member of Qatar Foundation).

X. Zhai is with the Department of Electronics, Computing and Mathematics, University of Derby, Derby, DE22 1GB, U.K. (e-mail: x.zhai@derby.ac.uk).

A. Ait-Si-Ali, A. Amira and F. Bensaali are with the KINDI Center for Computing Research, Qatar University, Qatar. (e-mail: amine.aitsiali@qu.edu.qa; abbes.amira@qu.edu.qa; f.bensaali@qu.edu.qa)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2016.2619181, IEEE Access

TABLE I. SOFTWARE AND HARDWARE BASED GAS IDENTIFICATION SYSTEMS

Papers	No. Sensors	Target gases	Pre-processing	Classification	Implementation
[4]	8	CO, H_2, CH_4, CO-H_2 & CO-CH_4	EN & PCA	Committee machine: KNNs, MLP, RBF, GMM & PPCA	FPGA Celoxica RC203
[5]	8	CO, H_2, CH_4, CO-H_2 & CO-CH_4	EN	MLP	FPGA APS-X208
[6]	16	CO, H_2, CH_4 & C_2H_5OH	SOM	IM & LDA	PC
[7]	8	O_3, LPG/LNG, NO_X , Alcohol, Smoke, VOC, CO& NH_3	SMA , Normalization between 0 and 1 & PCA	GA & ANN	Laptop, Zigbee & phone
[8]	8	CO, H_2, CH_4, CO-H_2 & CO-CH_4	PCA vs LDA vs NS	Density models: KNNs, GMM & GTM Discriminant functions: RBF, MLP and GLM	PC
[9, 10]	16	CO, H_2 & C_2H_6O	LSTE	RO	ASIC
[11]	16	CO, H_2 & C_2H_6O	With and without PCA	DT	FPGA (Xilinx Virtex II)+ ASIC
[12]	7 & 16	CO, H_2, C_2H_6O, CO_2, NH_3 & C_3H_8	With and without PCA & LDA	DT	PC & Zynq SoC

the sensor data in real-time and to provide a sensible classifier output as fast as possible, which is a must in failure detection systems. In this paper an overview of the proposed system approach and an optimised hardware implementation of a feed-forward MLP neural network are detailed. The key features of the proposed work are optimizations to have a fixed-point parallel MLP system for the system level integration of the system. This paper aims to present a hardware implementation of a MLP classifier starting with some background work followed by the system description which includes neural network architectures and MLP classifier design. Then the details of FPGA implementation of an MLP algorithm are presented. Finally conclusions about the approached design and future work are included.

II. Related Work

The gas classification problem has been widely addressed in the literature. A summary of various gas identification systems is presented in Table I, a comparison is made in terms of number of gas sensors used, target gases, pre-processing and classification algorithms as well as on the implementation platform. The pre-processing algorithms and classification algorithms used are the following: Euclidean normalization (EN), principal component analysis (PCA), linear discriminant analysis (LDA), neuroscale (NS), self-organized map (SOM), smoothed moving average (SMA), logarithmic spike timing encoding (LSTE), rank order (RO), *k*-nearest neighbors (KNNs), MLP, radial basis function (RBF), Gaussian mixture model (GMM), probabilistic principal component analysis (PPCA), image moment (IM), genetic algorithm (GA), ANN, generative topographic mapping (GTM), binary decision tree (DT) classifier and general linear model (GLM). The types of implementation platforms used are mainly personal computer (PC), field-programmable gate array (FPGA), application-specific integrated circuit (ASIC) and Zynq SoC.

In open literature many research papers have also been published about the use of FPGAs as the implementation platform for the MLP neural network for different applications. In [13], vitabile et al. implemented a MLP ANN that featured a virtual neuron based architecture with a target to have an optimized structure for high classification rate and minimum resource usage. The developed architecture was applied on high energy physics and road sign recognition algorithms. In [14], Yilmaz et al. implemented differential evaluation algorithm on an FPGA platform and cross compared with software simulations done in MATLAB. In [15], Alizadeh et al. implemented a ANN system that predicts cetane number in diesel fuel from the chemical compositions of the fuel by using the data from chromatography (LC) and gas chromatography (GC). In [16], Shi et al. implemented a gas discrimination system with the use of different classifiers on an FPGA platform. MLP was one of the classification algorithm which was implemented with the use of the tin-oxide gas sensors. In [17], Latino et al. implemented a memory based MLP architecture on an FPGA platform to work with a smart position sensor system.

In [18], Moradi et al. implemented an MLP for Farsi handwritten digit recognition algorithm on an FPGA platform

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2016.2619181, IEEE Access

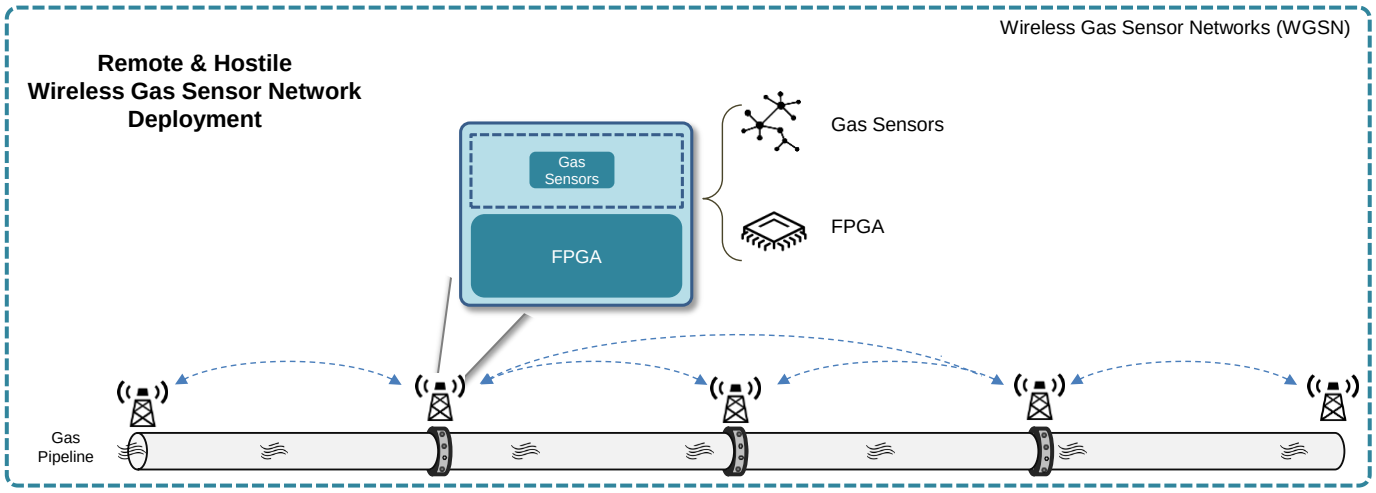

Fig. 1. Proposed system environment.

for speeding up an offline classification. In [19], Blaiech et al. proposed an efficient implementation of MLP neural network. Proposed a methodology to increase the efficiency of the implementation in terms of area and time, via an automatic generation of the MLP encoding. In [20], Ferreira et al. proposed an MLP architecture suitable to reduce the size of large ANN structures. The design is compared with software implementations. In [21], Bahoura et al. proposed a pipelined implementation to reduce the critical path and to increase the frequency of the design for an non-linear approximation ANN.

III. System Description

The proposed system environment and its main component are illustrated in Fig. 1. The proposed system consists of a series of gas sensor nodes, where each of this node is equipped with a gas sensor acquisition unit and a FPGA unit. The gas sensor acquisition unit is used to monitor the specific gas molecules, and the FPGA unit is used to analyse and classify the gas categories. The processed information will then be sent to the monitoring centre through the WGSN using a multichip based approach.

The block diagram of the proposed sensor node is shown in Fig. 2. The Overall system consists of a gas chamber where the gas sensor is present, data acquisition RFID transmitter and receivers, MLP classifier implemented on a ZYNQ platform, and a screen to show the system's output. The sensor data is read from the gas chamber using the data acquisition block and transmitted to the processing platform through RFID [22]. In this particular application the Processing System (PS) is used to handle communication and control of the RFID block as well as the MLP classifier working on the PL.

The final output of the overall system is the classified gas type from the trained neural network. The list of gases used during the data collection and training is:

- Benzene − C_6H_6
- CarbonMonoxide − CO
- Formaldehyde − CH_2O
- NitrogenDioxide − NO_2
- SulfurDioxide − SO_2

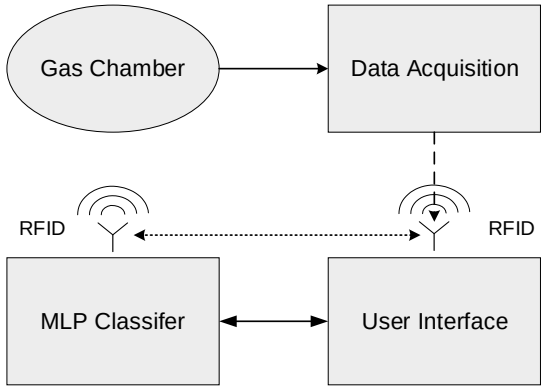

Fig. 2. Proposed sensor node architecture.

A. Neural Network Architecture

In this section, the notation, the ANN architecture and training process are explained. Neural networks have been widely used in pattern classification, completion, approximation, prediction and optimizations.

The MLP is an ANN that consists of multiple layers of neurons in a feed-forward architecture. A multilayer perceptron consists of three or more layers where one input, one output and one or more hidden layers. MLP uses nonlinear activation function with the neurons and each layer is fully connected to the next layer. Several perceptrons are combined in order to create decision boundary with the use of non-linear/linear activation functions. Each perceptron provides a non-linear mapping to a new dimension. Given that MLP is a fully connected network each neuron in each layer is connection to the next layer with a certain weight function w_{ij}. MLP uses a supervised learning technique called back propagation. During the training phase of the neural network

weight function are defined. The training method for the MLP is based on the minimisation of the chosen cost function which is initially developed by Werbos [23] and Parker [24].

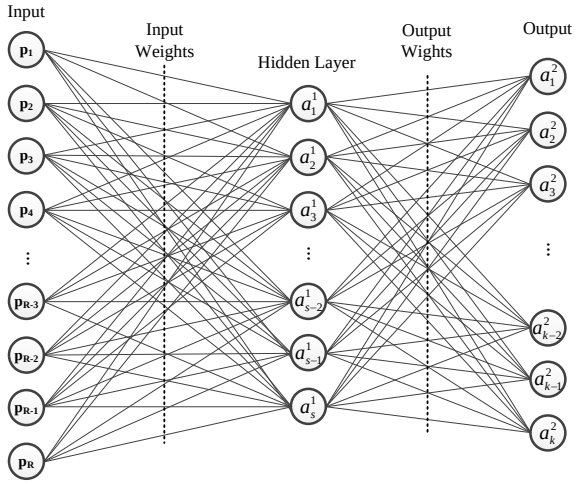

Fig. 3. The architecture of the two-layer feed-forward network.

Fig. 3 is an example of an MLP network, where a hidden layer consists of S neurons and each neuron has R weights, which can be presented in a $S \times R$ matrix called Input Weight matrix $\mathbf{I}$ as shown in equation 1. The input vector $\mathbf{P}$ has R elements $[p_1, p_2, \ldots, p_R]^T$, which are multiplied by $\mathbf{I}$ and the resulting matrix is summed with a bias vector $\mathbf{b_1}$ to form vector $\mathbf{n_1}$ as shown in equation 2. The output of the hidden layer $\mathbf{a_1}$ is the result of applying the transfer function on $\mathbf{n_1}$ (see equation 3).

$$\mathbf{I} = \begin{bmatrix} w_{1,1} & w_{1,2} & \ldots & w_{1,R} \\ w_{2,1} & w_{2,2} & \ldots & w_{2,R} \\ & \ldots\ldots & & \\ w_{S,1} & w_{S,2} & \ldots & w_{S,R} \end{bmatrix} \quad (1)$$

$$\mathbf{n_1} = \mathbf{I} \cdot \mathbf{p} + \mathbf{b_1} \quad (2)$$

$$\mathbf{a_1} = f_1(\mathbf{n_1}) \quad (3)$$

The same operations applied in the hidden layer are used in the output layer, which consists of K neurons, where $\mathbf{a_1}$ is used as the input vector (see equations 4, 5 and 6).

$$\mathbf{L} = \begin{bmatrix} w_{1,1} & w_{1,2} & \ldots & w_{1,S} \\ w_{2,1} & w_{2,2} & \ldots & w_{2,S} \\ & \ldots\ldots & & \\ w_{K,1} & w_{K,2} & \ldots & w_{K,S} \end{bmatrix} \quad (4)$$

$$\mathbf{n_2} = \mathbf{L} \cdot \mathbf{a_1} + \mathbf{b_2} \quad (5)$$

$$\mathbf{a_2} = f_2(\mathbf{n_2}) \quad (6)$$

B. MLP Classifier

The MLP algorithm which is a feed-forward ANN is modelled with two hidden layers and trained using the provided database [25]. The proposed feed-forward artificial neural network has 12 input neurons, three hidden neurons and one output neuron, where the 12 input neurons are corresponding to the 12 features extracted from each gas sample in a database that contains 600 gas samples. The neural network is trained by Levenberg-Marquardt backpropagation algorithm [26]. Fig. 4 shows the training performance of the ANN. This particular training uses 70% of the data for training, 15% of the dataset for testing and the remaining 15% for the validation. In general the hardware implementation of ANN system can hit to the PL's routing capabilities quickly since it requires layer based connectivity. Thus operations and ANN structure are implemented in optimized way that utilizes PL's routing and in fixed-point to limit the internal numerical precision which becomes a trade-off between hardware resources, calculation time and approximation quality.

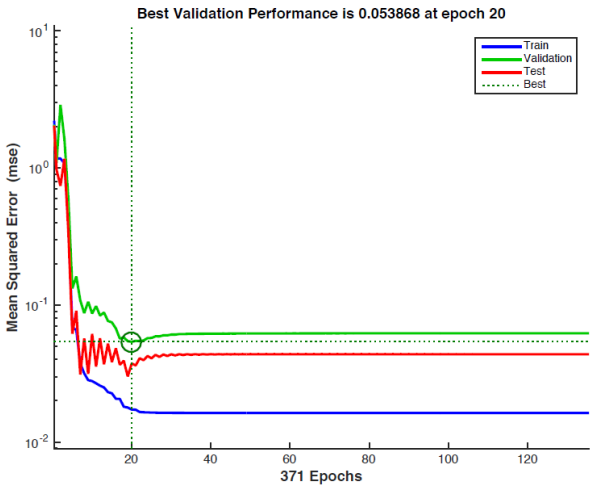

Fig. 4. ANN training performance.

The choice of the number of hidden layers is based on the performance of the ANN over the testing of using different hidden layers. Given the desired output of the ANN system is an integer which indicates the classified gas type, ANN with higher number of hidden layers tends to have very good performance during the training however they have bad performance with the validation stage. The optimal point for the ANN is chose where validation stage has the lowest Mean Square Error (MSE) value.

IV. HARDWARE IMPLEMENTATION

The MLP ANN training phase was done in MATLAB simulation environment. Since the data set and the gas sensors are not changing dynamically. Training is proposed to be done by recorded data and according to the training results. The implementation of the trained weight data is done using signed fixed-point representation 24-bit total length with 20 fractional bits. The activation function which has been used within the MLP algorithm is implemented with the use of look-up tables (LUTs). Different variations have been performed to obtain the optimum implementation for this component, however in order to reduce the latency, the distributed memory is chosen. The proposed parallel design can be described as a parallel data-flow architecture, where every neuron in every layer has one processing element (PE). This approach allows the system

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2016.2619181, IEEE Access

to work in parallel and to produce a high throughput system as shown in Fig. 5.

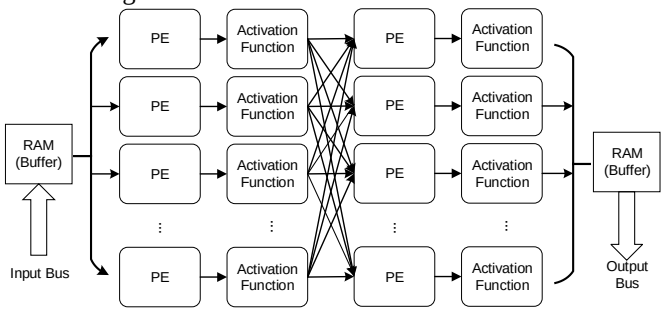

Fig. 5. FPGA implementation.

In Fig. 5, each PE consists of a RAM to store the weights of the neurons, where the input vector is firstly multiplied by the weights and accumulated to be fed into the activation function block. Fig. 6 shows the architecture of the PE.

The pseudo code of an input layer can be written as presented in Code 1.

Code 1: Pseudo code of an input layer

```
1. Input: P_in is the input data array of sensor readings.
2. Output: N_out is the output data array of the input layer.
3. for (row = 0; row < S; row++){
4.     for (col = 0; col < R; col++){
5.         N_out[row][col] += weights[row][col]× P_in[col];
6.     }
7.     N_out[row][col] = N_out[row][col] + b[row];
8. }
```

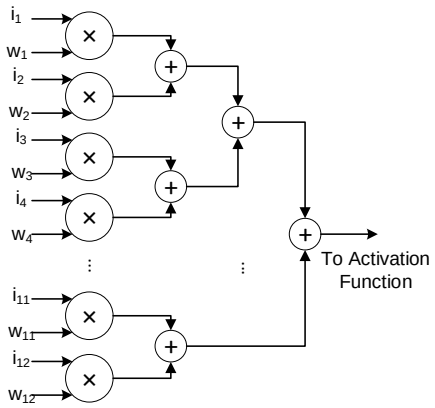

Fig. 6. Architecture of the processing element.

In the feedforward computation perspective, the multi-accumulation operations will occupy the most of the computation time. So we will focus on accelerating input and hidden layers. The objective of computation optimisation is to enable efficient loop unrolling or pipelining while fully utilisation of all computational resources provided by the FPGA on-chip hardware. The used optimisation pragmas are highlighted as follow:

Loop Unrolling: The loop unrolling strategy can be used to increase the utilisation of massive computation resources in FPGA devices. Depending on the way to unroll along different loop dimensions, the implementation variants will be generated. The complexity of the generated hardware will be affected by the data dependency of the unrolled execution instances as well as the affection from the unrolled hardware copies and the hardware operation frequency. An independent data sharing relation generates direct connections between buffers and computation elements. However, a dependent data sharing relation generates interconnects with complex hardware topology. Loop dimension *col* is selected to be unrolled to avoid complex connection topologies for all arrays.

Loop Pipelining: loop pipelining is another key optimisation techniques in high-level synthesis to improve the throughput of the system, where the execution of operations from different loop iterations are overlapped in an organised way. Similar to the loop unrolling techniques, the throughput achieved is limited by resource constraints and data dependency in the loop. Code structure after optimisation for loop unrolling and loop pipelining is shown in Code 2.

Code 2: Pseudo code of an input layer (optimized structure)

```
9.  Input: P_in is the input data array of sensor readings.
10. Output: N_out is the output data array of the input layer.
11. for (row = 0; row < S; row++){
12. #pragma HLS pipeline
13.     for (col = 0; col < R; col++){
14. #pragma HLS UNROLL
15.         N_out[row][col] += weights[row][col]× P_in[col];
16.     }
17.     N_out[row][col] = N_out[row][col] + b[row];
18. }
```

Within the MLP algorithm there are several activation functions, linear and non-linear, which can be chosen according to the application domains behaviour. In this implementation the activation function is used as sigmoid. For the actual hardware implementation the sigmoid function shown in Fig. 7, is chosen to be represented as signed fixed-point representation was limited in order to accomplish the hardware implementation with a feasible resource usage, where each value is implemented in 16-bit total length with 14 fractional bits.

The Tan-sigmoid function shown in equation 7 is implemented using its simplified version (i.e. equation 8). When $x < -5$ or $x > +5$, the values of $tanh(x)$ is closed to -1 and +1 respectively. When $-5 \leq x \leq +5$, the values of $tanh(x)$ have been pre-calculated for using samples of x in this range {-5, -4.99,…,4.99,5} where a step size 0.01 is used.

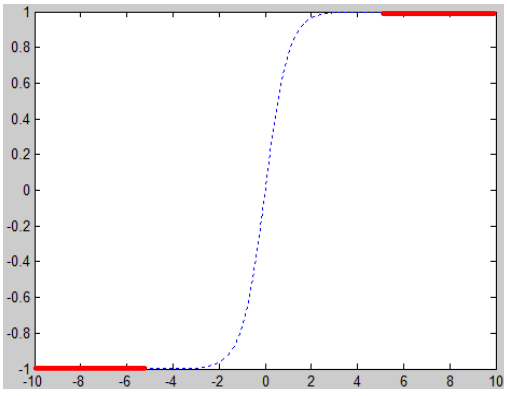

Fig. 7. Sigmoid Function.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2016.2619181, IEEE Access

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{7}$$

$$\tanh(x) = \begin{cases} -1 & (x < -5) \\ \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}} & (-5 \le x \le +5,\ stepsize: 0.01) \\ +1 & (x > +5) \end{cases} \tag{8}$$

The results from equation 8 are pre-calculated and stored in a memory. Since the used step size is 0.01 for the range -5 to +5, there are 1001 results to be stored in the memory. In order to access the correct pre-calculated results from the memory, the following formula needs to be used to calculate the address:

$$address = (x + 5) \times 100 \tag{9}$$

The entire ROM-based Tan-sigmoid block is shown in Fig. 8. a_1 is a register used to store the current result from Tan-sigmoid block, which represents one element from the vector a_1 in equation 3.

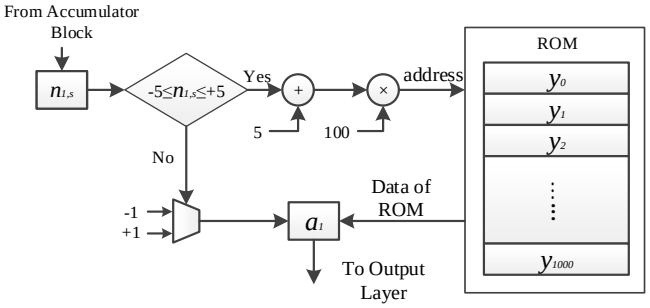

Fig. 8. ROM-based Tan-sigmoid.

The hidden layer of the MLP is implemented using similar way of the input layer, which is responsible to gather classification output data from the MLP and send the response to the PS to be used by the user. An overview of the implementation is shown in Fig. 9. The whole system fits in one Zynq SoC chip and uses a DDR3 DRAM and SD card for external storage. ARM Cortex-A9, a hard processor core is used to assist with MLP accelerator start-up, reading and storing data from/to SD card. AXI4lite bus is used for transmitting the weights, input/output data between PS/PL. The MLP accelerator works as an IP on the AXI4 bus, it receives configuration parameters from ARM Cortex-A9 through AXI4lite bus and sends the processed the data back. Interrupt mechanism is enabled between ARM processor and MLP accelerator to provide an accurate time measurement.

V. Performance Evaluation

Although the proposed FPGA implementation of MLP is mainly based on fixed-point arithmetic, the accuracy of the proposed implementation is not compromised, and it has the same results as the floating-point implementation in MATLAB. The major benefit of using fixed-point implementation is to reduce the hardware cost and optimize the system in terms of speed and latency. The accelerator design is implemented with Vivado HLS (v2016.1). This tool enables implementing the MLP accelerator using C/C++ language and C/RTL simulation as well as exporting the RTL as a Vivado's IP core. The C++ code of the MLP design is parallelized by adding HLS-defined pragma and the pre-synthesis parallel version is validated with the C/RTL co-simulation tool. The pre-synthesis resource report are used for design resource exploration and performance estimation. The RTL is exported as IP core to be synthesised and implemented in Vivado (v2016.1). The trained MLP neuron network is implemented on the Zybo board which has a Xilinx Zynq-7000 XC7Z010T-1CLG400 all programmable SoC. The working frequency of PS and PL is 650 MHz and 100 MHz respectively. The software implementation runs on a dual-core Intel i7-5600U CPU at 2.6 GHz.

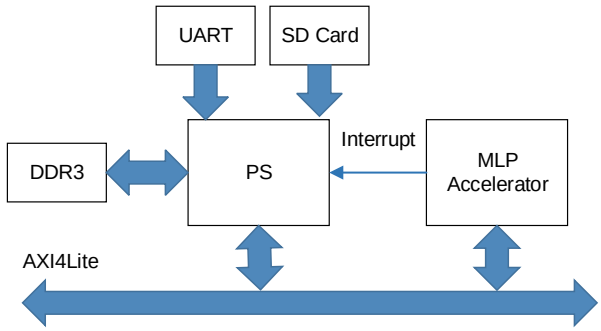

Fig. 9. Implementation overview.

A. Experimental Results

In this subsection, the resource utilisation is firstly reported. Subsequently, the software implementation (on CPU) and the implementation of the proposed MLP accelerator on FPGA is compared. Finally, the comparison of the proposed implementation and existing FPGA approaches is provided.

The Vivado tool is used to complete the placement and routing of the proposed implementation. The resources utilisation of the proposed implementation is reported out, as shown in Table II.

TABLE II. FPGA Resource Utilisation

Resource	DSP	BRAM	FF	LUT
Used	28	2	2863	4032
Available	80	60	35200	17600
Utilisation (%)	35	3.33	8.13	22.91

As it can be seen from Table II, the proposed implementation has only consumed 29.98% slice of the available PL resources. The 35% DSP48E1 is used to implement multipliers within the input and hidden layers. Since all loops (Code 2) have been optimised either using parallel or pipeline structures, the usage of DSPs is dominant. In order to maximise the performance of the proposed implementation, the input weight array **I** and input vector array **P** shown in equation 2 have been repartitioned into $S \times R$

and *R* smaller arrays respectively, where *S* and *R* is the number of hidden and input neurons respectively. The major benefit of doing this is to ensure all the data within the arrays can be fed into the parallelised multipliers, to be executed at the same time. The achieved synthesised architecture confirmed that the desired initiation interval for the pipeline pragma achieves one initiation, which has greatly reduced data dependencies of input arrays.

During the conversion process several criteria can be considered, such as MSE, size of neural network, memory requirements. The absolute error calculated by the difference between success rates in fixed-point coding and floating-point coding reflects the performance criterion of our MLP. During the fixed-point conversion process, we determined the integer and fractional part of the MLP after the arithmetic conversion, which took the floating point double precision and converted to 2's complement with 24 bit total length with 20 fractions. As a result of this, the performance of classifier with fixed-point implementation remained same as 96.8% correct classifications from the MATLAB implementation.

The ARM processor runs at 650 MHz and the PL clocked at 100 MHz. The processing time of the proposed system is measured by counting the number of ARM processor's clock cycles spent for obtaining the classified results of one Gas signal from the MLP accelerator. Table III shows the comparison between the software and hardware implementations of the MLP accelerator in terms of the processing time. As it can be seen in Table III, the average processing time using the hardware implementation has improved by a factor of 31 compared to the software implementation on a dual-core Intel i7-5600U CPU at 2.6 GHz.

TABLE III. PROCESSING TIME OF THE MLP ACCELERATOR

	Hardware Implementation (us)	PC Software Implementation (us)	Factor
Processing time	0.5397	17	31

TABLE IV. ESTIMATION OF POWER CONSUMPTION

	Utilization Details	Power (W)	Utilization (%)
Dynamic Power Consumption	Clock	0.009	1
	Signals	0.025	2
	Logic	0.019	1
	DSP	0.028	2
	BRAM	0.004	<1
	PS7	1.556	93
Static Power Consumption	Device Static	0.135	8

The on-chip power consumption consists mainly of two parts, which are static and dynamic power consumption. The static power is consumed due to transistor leakage. The dynamic power is consumed by fluctuating power as the design runs, i.e. Zynq7 Processing System (PS7), clock, power, logic power, signal power, BRAMs power, etc., which are directly affected by the chip clock frequency and the usage of chip area. The details of estimated power consumption of the implementation are summarised in Table IV. The PS7 consumes much more power than the PL; this is due to the fact that the ARM dual core Cortex-A9 based processing system has much higher running frequency than the PL and it runs drivers and control programmes. Compare to the PS7, the custom logic blocks consumes only a small portion of the total on-chip power consumption.

To measure the performance of MLP neural network across the wide range of platforms that they have been implemented on, one of the common criteria is: connections per second (CPS), where the connections stand for one term in the sum of

$$\sum_j w_{S,R} p_R \tag{10}$$

where $w_{S,R}$ and p_R are the elements within Input Weight matrix **I** and input vector **P** respectively. Due to the parallel implementation of FPGA based neural networks, the CPS metric increases as long with the number of synaptic connections. While taking account the precision of the network, the connection primitives per second (CPPS) is calculated, which is equal to $b_x \times b_w \times$ CPS, where b_x and b_w are the numbers of bit for the inputs and its weights. While taking account of usage of resources, we calculate the CPPS per LUT (CPPSL). Table V shows an implementation comparison between the proposed network and other implementations using these metrics.

The CPPSL metric is a means for normalising the throughput by dividing by the number of LUTs used in the implementation. The implementation of the proposed system maximises flexibility as well as the parallelism over hardware cost. The architecture of hardware and software co-design allows the weights of the ANN on the PL site to be easily updated based on the inputs from the PS site.

TABLE V. A COMPARISON OF ANN FPGA IMPLEMENTATIONS

	Platform	LUT	DSP48Es	ANN size	CPS	CPPSL
Proposed work	XC7Z010T	4032	28	12-3-1	72.3M	10.3M
[27]	XC5VSX50T	8043	70	10-3-1	536M	9.6M
[28]	Virtex-4 LX40	4346	8	748-50-34	1.2M	0.07M
[29]	Virtex-4LX25	7748	27	N/A	N/A	N/A

Table VI contains a comparison of existing FPGA-based gas classification implementations. Unlike the other existing FPGA-based gas classifiers, the proposed work is implemented on a low-cost Zynq SoC and achieved a similar recognition rate and processing speed. Although, the work presented in [12] shows outstanding results in terms of classification rates and resources utilisations, however, it is

TABLE VI. A COMPARISON OF EXISTING FPGA-BASED GAS CLASSIFICATION IMPLEMENTATIONS

Works	No. sensor arrays	Dimensionality reduction algorithms	Classification algorithms	Classification accuracy (%)	Execution time (ns)	Implementation platform	Hardware resources			
							BRAM	DSP48E	FF	LUT
[12]	16	PCA	DT	91.7	795	Zynq SoC ZC702	0	20	2537	6191
		LDA		95.0	746		0	5	958	1732
	7	PCA		100	410		0	10	1533	3213
		LDA		100	360		0	5	755	1629
[30]	16	PCA	KNN	96.7	3073		3	67	8259	12754
This work	12	-	MLP	97.4	540	Zynq SoC ZYBO	2	28	2863	4032

worth noting that the proposed work uses a larger sensor array and it does not need any pre-calculation on the parameters from a PC to assist the calculation on FPGA, which means that the proposed is a complete implementation that can be used as a standalone platform to classify the gas in a low-cost and efficient way.

VI. CONCLUSIONS

In this paper, a parallel high speed MLP has been successfully implemented on the ZYNQ SoC, to work with a gas sensor array system. The MLP was trained using a dataset that has been recorded with a gas chamber. In the implementation of the trained back-propagation MLP algorithm, 50% of the 196 inputs are used as training set for training phase and 25% of inputs as test and 25% for the validation.

The proposed system has been implemented on the Xilinx Zynq-7000 XC7Z010T-1CLG400 from Xilinx. The Vivado 2016.1 has been used to implement and synthesise the implementation. Compared to the other implementations in the open literature, the proposed approach uses two hidden layers with 12 input neurons, 3 hidden neurons and 1 output neuron. In this work optimizing methodology for the implementation of an MLP neural network for gas applications has been performed which allowed us to reduce the requirement of the word-length also the complexity of the ANN system, number of hidden layers. This methodology has helped to optimize the response of the MLP.

In terms of future work, it is to develop an intelligent algorithm for sensor failure detection and self-recovery from the fault status. In addition to this, investigating data fusion techniques for data transmission optimisation over the WGSN would be other key research direction for this project.

ACKNOWLEDGMENT

We would also like to thank Prof. Amine Bermak (Hamad bin Khalifa University, Qatar) for providing the data used in this paper. The statements made herein are solely the responsibility of the authors.

REFERENCES

[1] A. Sajid, H. Abbas, and K. Saleem, "Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges," IEEE Access, vol. 4, pp. 1375-1384, 2016.

[2] M. A. Akbar, A. A. S. Ali, A. Amira, M. Benammar, F. Bensaali, S. Mohamad, et al., "A multi-sensing reconfigurable platform for gas applications," in 2014 26th International Conference on Microelectronics (ICM), 2014, pp. 148-151.

[3] A. K. Jain, R. P. W. Duin, and M. Jianchang, "Statistical pattern recognition: a review," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, pp. 4-37, 2000.

[4] M. Shi, A. Bermak, S. Chandrasekaran, A. Amira, and S. Brahim-Belhouari, "A Committee Machine Gas Identification System Based on Dynamically Reconfigurable FPGA," IEEE Sensors Journal, vol. 8, pp. 403-414, 2008.

[5] F. Benrekia, M. Attari, and M. Bouhedda, "Gas Sensors Characterization and Multilayer Perceptron (MLP) Hardware Implementation for Gas Identification Using a Field Programmable Gate Array (FPGA)," Sensors vol. 13, pp. 2967-2985.

[6] A. B. Far, F. Flitti, B. Guo, and A. Bermak, "A Bio-Inspired Pattern Recognition System for Tin-Oxide Gas Sensor Applications," IEEE Sensors Journal, vol. 9, pp. 713-722, 2009.

[7] E. Kim, S. Lee, J. H. Kim, C. Kim, Y. T. Byun, H. S. Kim, et al., "Pattern recognition for selective odor detection with gas sensor arrays," Sensors, vol. 12, pp. 16262-16273, 2012.

[8] S. B. Belhouari, A. Bermak, G. Wei, and P. Chan, "Gas identification algorithms for microelectronic gas sensor," in Proceedings of the 21st IEEE Instrumentation and Measurement Technology Conference, 2004, pp. 584-587.

[9] K. T. Ng, B. Guo, A. Bermak, D. Martinez, and F. Boussaid, "Characterization of a logarithmic spike timing encoding scheme for a 4× 4 tin oxide gas sensor array," in Sensors, 2009 IEEE, 2009, pp. 731-734.

[10] K. Ting Ng, B. Guo, D. Martinez, F. Boussaid, and A. Bermak, "A 4× 4 tin oxide gas sensor array based on spike sequence matching," in 2nd International Conference on Signals, Circuits and Systems, 2008.

[11] Q. Li and A. Bermak, "A low-power hardware-friendly binary decision tree classifier for gas identification," Journal of Low Power Electronics and Applications, vol. 1, pp. 45-58, 2011.

[12] M. A. Akbar, A. A. S. Ali, A. Amira, F. Bensaali, M. Benammar, M. Hassan, et al., "An Empirical Study for PCA and LDA Based Feature Reduction for Gas Identification," IEEE Sensors Journal

[13] S. Vitabile, V. Conti, F. Gennaro, and F. Sorbello, "Efficient MLP digital implementation on FPGA," in Proceedings of the 8th Euromicro Conference on Digital System Design, 2005. , 2005, pp. 218-222.

[14] A. R. Yilmaz, B. Erkmen, and O. Yavuz, "FPGA implementation of Differential Evaluation Algorithm for MLP training," in 2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings, 2014, pp. 425-430.

[15] G. Alizadeh, J. Frounchi, M. B. Nia, M. H. Zarifi, and S. Asgarifar, "An FPGA implementation of an Artificial Neural Network for prediction of cetane number," in International Conference on Computer and Communication Engineering, 2008., 2008, pp. 605-608.

[16] M. Shi, S. Chandrasekaran, A. Bermak, and A. Amira, "FPGA Based Run Time Reconfigurable Gas Discrimination System," in International Symposium on Integrated Circuits, 2007, 2007, pp. 180-183.

[17] C. Latino, M. A. Moreno-Armendariz, and M. Hagan, "Realizing general MLP networks with minimal FPGA resources," in International Joint Conference on Neural Networks, 2009, pp. 1722-1729.
[18] M. Moradi, M. A. Poormina, and F. Razzazi, "FPGA Implementation of Feature Extraction and MLP Neural Network Classifier for Farsi Handwritten Digit Recognition," in Third UKSim European Symposium on Computer Modeling and Simulation, 2009, pp. 231-234.
[19] A. G. Blaiech, K. B. Khalifa, M. Boubaker, and M. H. Bedoui, "Multi-width fixed-point coding based on reprogrammable hardware implementation of a multi-layer perceptron neural network for alertness classification," in 10th International Conference on Intelligent Systems Design and Applications (ISDA), 2010, pp. 610-614.
[20] A. P. d. A. Ferreira and E. N. d. S. Barros, "A high performance full pipelined arquitecture of MLP Neural Networks in FPGA," in 17th IEEE International Conference on Electronics, Circuits, and Systems (ICECS), 2010, pp. 742-745.
[21] M. Bahoura and C. W. Park, "FPGA-implementation of high-speed MLP neural network," in 18th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2011, pp. 426-429.
[22] B. Guo, A. Bermak, P. C. H. Chan, and G. Z. Yan, "An Integrated Surface Micromachined Convex Microhotplate Structure for Tin Oxide Gas Sensor Array," IEEE Sensors Journal, vol. 7, pp. 1720-1726, 2007.
[23] P. J. Werbos, "Backpropagation through time: what it does and how to do it," Proceedings of the IEEE, vol. 78, pp. 1550-1560, 1990.
[24] D. Parker, "Technical Report tr-47," Massachusetts Institute of Technology1985.
[25] M. Hassan, S. B. Belhaouari, and A. Bermak, "Probabilistic Rank Score Coding: A Robust Rank-Order Based Classifier for Electronic Nose Applications," IEEE Sensors Journal, vol. 15, pp. 3934-3946, 2015.
[26] G. Lera and M. Pinzolas, "Neighborhood based Levenberg-Marquardt algorithm for neural network training," IEEE Transactions on Neural Networks, vol. 13, pp. 1200-1203, 2002.
[27] A. Gomperts, A. Ukil, and F. Zurfluh, "Development and Implementation of Parameterized FPGA-Based General Purpose Neural Networks for Online Applications," IEEE Transactions on Industrial Informatics, vol. 7, pp. 78-89, 2011.
[28] X. Zhai, F. Bensaali, and R. Sotudeh, "Real-time optical character recognition on field programmable gate array for automatic number plate recognition system," IET Circuits, Devices & Systems, vol. 7, pp. 337-344, 2013.
[29] G. Alizadeh, J. Frounchi, M. B. Nia, M. H. Zarifi, and S. Asgarifar, "An FPGA implementation of an Artificial Neural Network for prediction of cetane number," in Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on, 2008, pp. 605-608.
[30] A. Ait Si Ali, A. Amira, F. Bensaali, Benammar, and A. M.; Bermak, "HW/SW Co-Design Based Implementation of Gas Discrimination," presented at the The 27th Annual IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP), London, 2016.